# FILES For Dummies

- by -
alix

*or how to find your way around the FILES statements without getting lost!*
This file was created on Mar 9, 2008 12:08 am

Are you one of those programmers that gets filled with dread each time they have to make use of the FILES statement? Do you think "Oh no, not that FILES statement again!" because you know it's going to take several trials and errors runs before you get it right? All you want is to retrieve some simple information about a specific directory, and then get on with the rest of your program...

Why suffer? Here is a painless way of remembering how to use the FILES statement so that you can get the information you need about files and directories with minimum frustration.

## Draw it!

Most of us understand that the FILES statement stores information in a double-dimensioned string array. FILES will also take care of adjusting the size of this array in case more space is needed. So we dutifully start our code with:

```
dim info$(10, 10)
```

Now what?

That's where the trouble starts...

1. First problem : Where does the all powerful FILES statement store its info?
2. Second Problem : How do we ask Mr. All Powerful to give us the info we need?

Note that problem 2 depends heavily on Problem 1 being solved.

This is where a simple drawing comes to the rescue.
Draw the info$() array as a table with rows and columns. Then, write each item of info in each cell :

| Info$(x,y) | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | **n=Number of Files** | **N=Number of Subdirectories** | **Drive** | **Path** |
| 1 | **Name of file 1** | **File Size** | **Date/Time** | **Attributes : r/h/s/a** |
| 2 | Name of file 2 | File Size | Date/Time | Attributes : r/h/s/a |
| ... | ... | ... | ... | ... |
| n | Name of last file | File Size | Date/Time | Attributes: r/h/s/a |
| n+1 | **Path + Name of Subdirectory1** | **Name of Subdirectory1** | | |
| n+2 | Path + Name of Subdirectory2 | Name of Subdirectory2 | | |
| .... | ... | ... | | |
| n+N | Path+Name of last subdir | Name of last subdir | | |

Et voilà! The next time you must use the FILES statement, have a look at the FILES table. Now you can quickly locate the info you need. It is as as easy as playing Bingo:

- Want to know in which drive is your folder? That's info$(0,2)
- How many subdirectories are there in your folder? That's info$(0,1)
- Name the second file in your folder? That's : info$(2,0)
- Is the third file in your folder read only? That's : check info$(3,3) if it contains an **r**

Note: r = readonly ; h = hidden ; s = system ; a = archive

# Examples :

To finish, let's have some simple examples.
In both examples I chose DefaultDir$ so that you can copy and paste my code, and run it without any problems.But, you could have something like:

```
myfolder$="C:\anyfolder"
```

It's also important that you dim an array to hold the retrieved information. This should be done in the beginning of your program. The FILES command will take care of redimming (resizing) the array as needed, but it must be defined before the FILES command is used. In these examples, the array is named info$()

```
dim info$(1, 1)
```

**How to list all the files contained in your folder:**

```
dim info$(1, 1)
```

```
myfolder$=DefaultDir$
FILES myfolder$, info$()
totfiles = val(info$(0,0))
for i=1 to totfiles
    name$= info$(i,0)
    print "-";name$
next
```

**How to list all the subdirectories contained in your folder:**

```
dim info$(1, 1)
myfolder$=DefaultDir$
FILES myfolder$, info$()
totfiles = val(info$(0,0))
totsubs = val(info$(0,1))
for i=totfiles+1 to totfiles+totsubs
    name$= info$(i,1)
    print "-";name$
next
```